Diabetes Analysis

In [27]:
```
#  Import Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [29]:
```
#  Load Dataset
df = pd.read_csv("diabetes.csv")
df
```

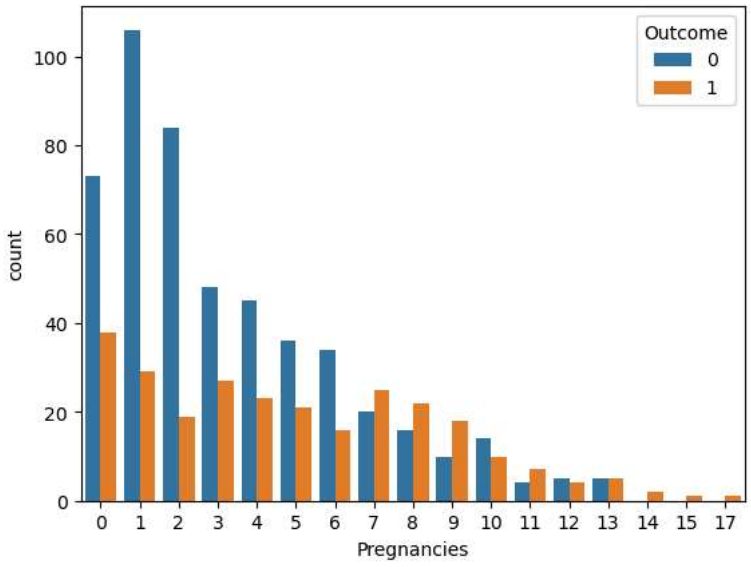
Out[29]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

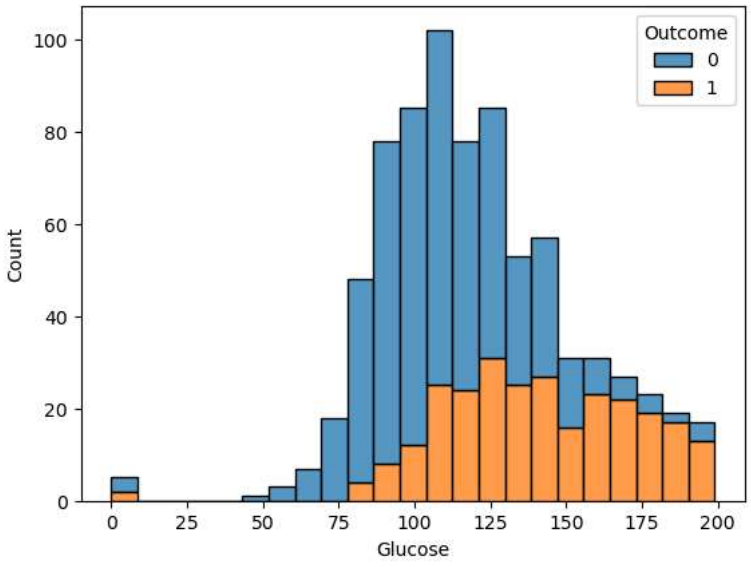
In [31]:
```
sns.countplot(data=df,x="Pregnancies",hue="Outcome")
```

Out[31]: <Axes: xlabel='Pregnancies', ylabel='count'>



In [35]:
```
sns.histplot(data=df,x="Glucose",hue="Outcome",multiple="stack")
```

Out[35]:
```
<Axes: xlabel='Glucose', ylabel='Count'>
```



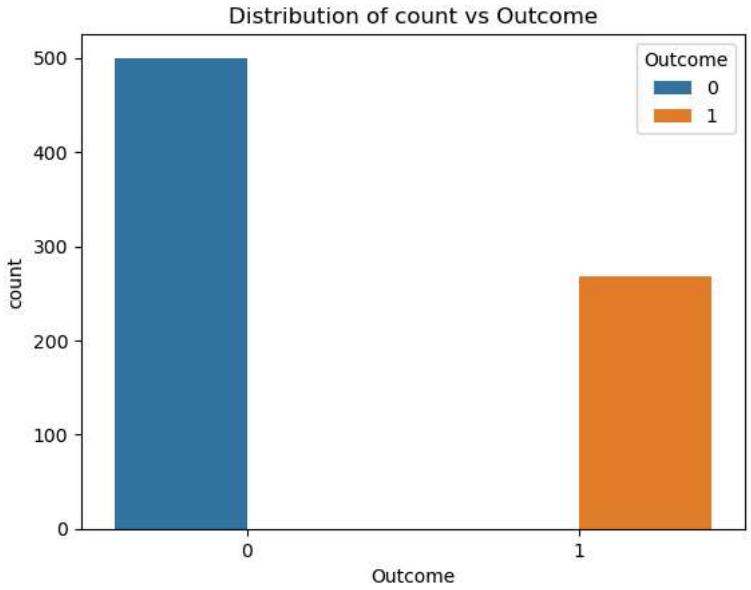
In [36]:
```
df.isnull().sum()
```

Out[36]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [37]:
```
copy(deep=True)
regnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']]=df_copy[['Pregnancies','
py.isnull().sum())
```

```
Pregnancies                 111
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```

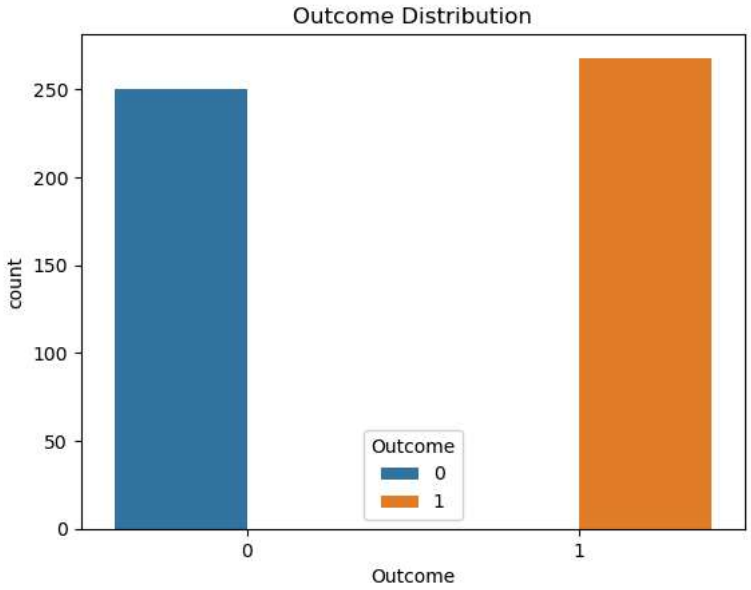
In [45]:
```
sns.countplot(data=df, x="Outcome", hue="Outcome")
plt.title("Distribution of count vs Outcome")
plt.show()

print(df['Outcome'].value_counts())
```




```
Outcome
0    500
1    268
Name: count, dtype: int64
```

In [39]:
```
from sklearn.utils import resample
df_majority=df[(df['Outcome']==1)]
df_minority=df[(df['Outcome']==0)]
df_minority_upsampled=resample(df_minority,n_samples=250,random_state=0)
df2=pd.concat([df_minority_upsampled,df_majority])
```

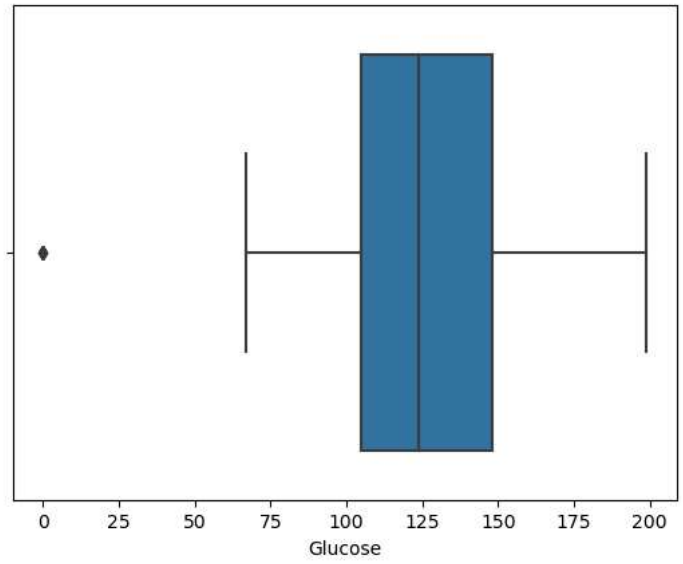
In [46]:
```
sns.countplot(data=df2, x="Outcome", hue="Outcome")
plt.title("Outcome Distribution")
plt.show()

print(df2['Outcome'].value_counts())
```




```
Outcome
1    268
0    250
Name: count, dtype: int64
```

In [42]:
```
sns.boxplot(x=df2["Glucose"])
```

Out[42]: <Axes: xlabel='Glucose'>

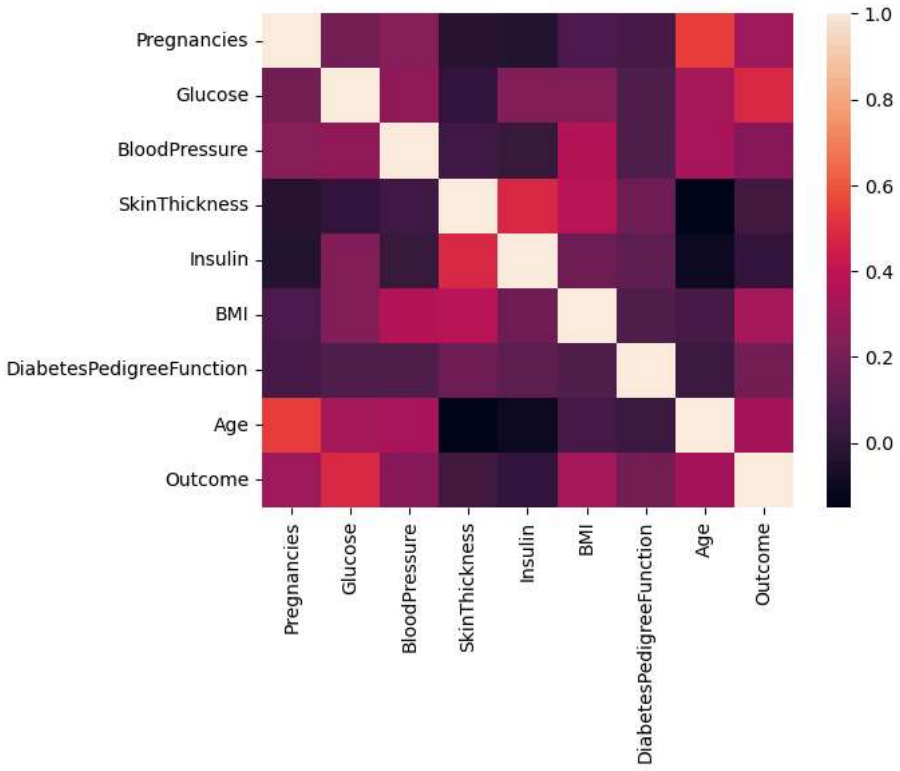


In [47]:
```
import scipy.stats as stats
z=np.abs(stats.zscore(df2))
data_clean=df2[(z<3).all(axis=1)]
data_clean.shape
```

Out[47]: (458, 9)

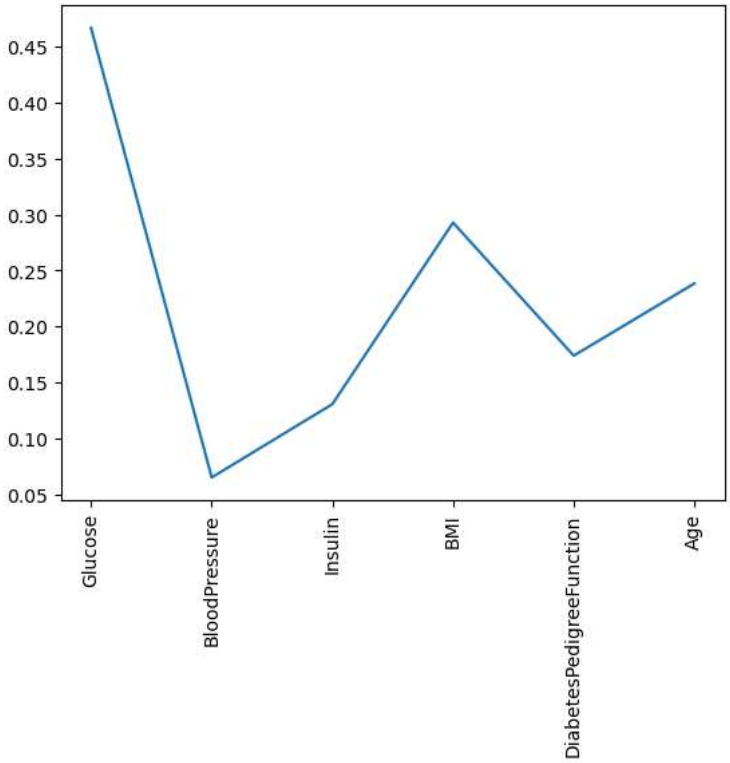
In [48]:
```
sns.heatmap(data_clean.corr(),fmt='.2g')
```

Out[48]: <Axes: >



In [49]:
```
data_clean2=df.drop(columns=['SkinThickness'])
```

In [50]:
```
corr=data_clean2[data_clean2.columns[1:]].corr()['Outcome'][:-1]
plt.plot(corr)
plt.xticks(rotation=90)
plt.show()
```



In [51]:
```
X=data_clean2.drop('Outcome',axis=1)
y=data_clean2['Outcome']
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)
```

In [52]:
```
#Random Forest

from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(random_state=0)
rfc.fit(X_train,y_train)
```

Out[52]:
```
RandomForestClassifier
RandomForestClassifier(random_state=0)
```

In [53]:
```
y_pred=rfc.predict(X_test)
print("Accuracy score:",round(accuracy_score(y_test,y_pred)*100,2),"%")
```

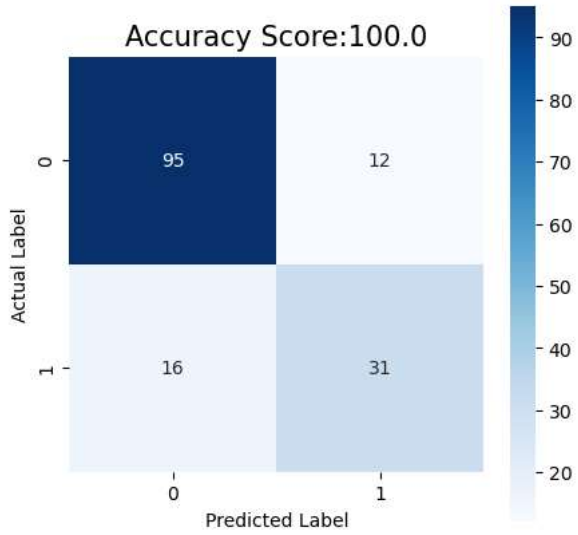
```
Accuracy score: 81.82 %
```

In [54]:
```
from sklearn.metrics import accuracy_score,precision_score,f1_score,recall_score
print("F1-score score:",(f1_score(y_test,y_pred)))
print("Recall-score score:",(recall_score(y_test,y_pred)))
print("Precision-score score:",(precision_score(y_test,y_pred)))
```

```
F1-score score: 0.6888888888888888
Recall-score score: 0.6595744680851063
Precision-score score: 0.7209302325581395
```

In [55]:
```
from sklearn.metrics import confusion_matrix,classification_report
cm=confusion_matrix(y_test,y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,annot=True,square=True,cmap='Blues')
plt.ylabel("Actual Label")
plt.xlabel("Predicted Label")
plt.title("Accuracy Score:100.0",fontsize=15)
```

Out[55]:
```
Text(0.5, 1.0, 'Accuracy Score:100.0')
```



In [56]:
```
#KNN


from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(X_train,y_train)
```

Out[56]:
```
KNeighborsClassifier
KNeighborsClassifier()
```

In [57]:
```
y_pred=knn.predict(X_test)
print("Accuracy score:",round(accuracy_score(y_test,y_pred)*100,2),"%")
```

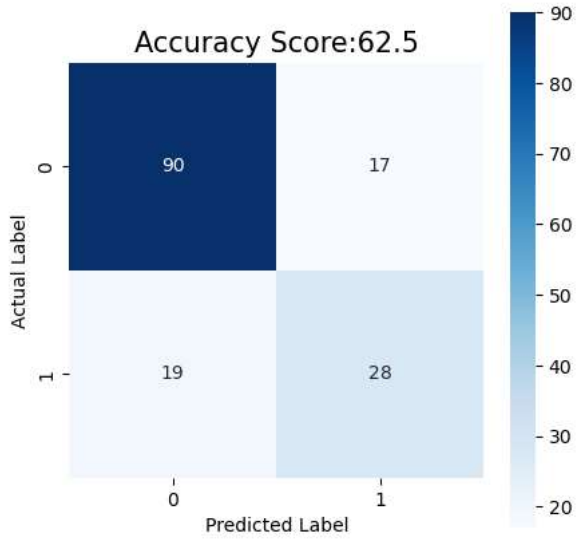
```
Accuracy score: 76.62 %
```

In [58]:
```
from sklearn.metrics import accuracy_score,precision_score,f1_score,recall_score
print("F1-score score:",(f1_score(y_test,y_pred)))
print("Recall-score score:",(recall_score(y_test,y_pred)))
print("Precision-score score:",(precision_score(y_test,y_pred)))
```

```
F1-score score: 0.608695652173913
Recall-score score: 0.5957446808510638
Precision-score score: 0.6222222222222222
```

In [59]:
```
from sklearn.metrics import confusion_matrix,classification_report
cm=confusion_matrix(y_test,y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,annot=True,square=True,cmap='Blues')
plt.ylabel("Actual Label")
plt.xlabel("Predicted Label")
plt.title("Accuracy Score:62.5",fontsize=15)
```

Out[59]:
```
Text(0.5, 1.0, 'Accuracy Score:62.5')
```



In [60]:
```
#AdaBooST


from sklearn.ensemble import AdaBoostClassifier
ada=AdaBoostClassifier(random_state=0)
ada.fit(X_train,y_train)
```

Out[60]:
```
AdaBoostClassifier(random_state=0)
```

In [61]:
```
y_pred=ada.predict(X_test)
print("Accuracy score:",round(accuracy_score(y_test,y_pred)*100,2),"%")
```

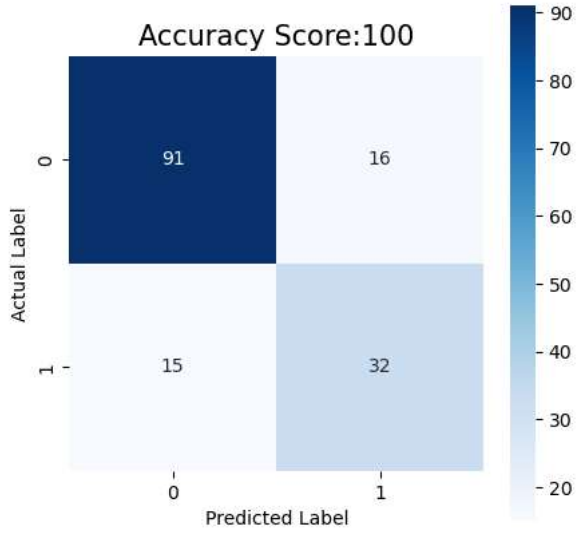
```
Accuracy score: 79.87 %
```

In [62]:
```
from sklearn.metrics import accuracy_score,precision_score,f1_score,recall_score
print("F1-score score:",(f1_score(y_test,y_pred)))
print("Recall-score score:",(recall_score(y_test,y_pred)))
print("Precision-score score:",(precision_score(y_test,y_pred)))
```

```
F1-score score: 0.6736842105263158
Recall-score score: 0.6808510638297872
Precision-score score: 0.6666666666666666
```

In [63]:
```
from sklearn.metrics import confusion_matrix,classification_report
cm=confusion_matrix(y_test,y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,annot=True,square=True,cmap='Blues')
plt.ylabel("Actual Label")
plt.xlabel("Predicted Label")
plt.title("Accuracy Score:100",fontsize=15)
```

Out[63]:
```
Text(0.5, 1.0, 'Accuracy Score:100')
```



In [64]:
```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(random_state=0)
lr.fit(X_train,y_train)
```

```
C:\Users\Administrator\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to c
onverge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/line
ar_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[64]:
```
LogisticRegression(random_state=0)
```

In [65]:
```
y_pred=lr.predict(X_test)
print("Accuracy score:",round(accuracy_score(y_test,y_pred)*100,2),"%")
```

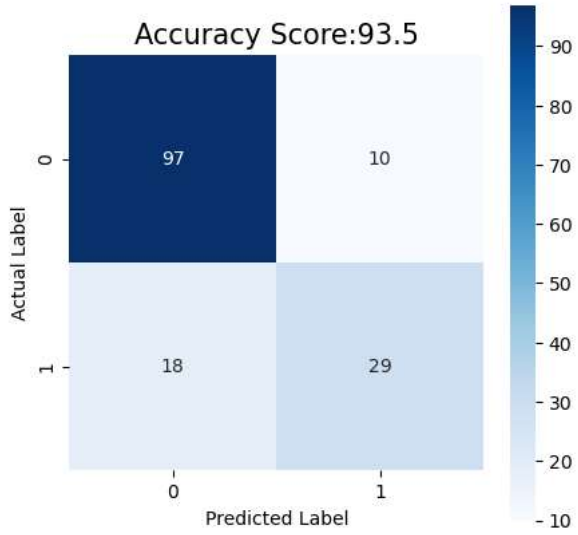
```
Accuracy score: 81.82 %
```

In [66]:
```
from sklearn.metrics import accuracy_score,precision_score,f1_score,recall_score
print("F1-score score:",(f1_score(y_test,y_pred)))
print("Recall-score score:",(recall_score(y_test,y_pred)))
print("Precision-score score:",(precision_score(y_test,y_pred)))
```

```
F1-score score: 0.6744186046511628
Recall-score score: 0.6170212765957447
Precision-score score: 0.7435897435897436
```

In [67]:
```
from sklearn.metrics import confusion_matrix,classification_report
cm=confusion_matrix(y_test,y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,annot=True,square=True,cmap='Blues')
plt.ylabel("Actual Label")
plt.xlabel("Predicted Label")
plt.title("Accuracy Score:93.5",fontsize=15)
```

Out[67]:
```
Text(0.5, 1.0, 'Accuracy Score:93.5')
```



In [ ]: